

日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出 願 年 月 日

Date of Application:

2002年 7月22日

出 願 番 号

Application Number:

特願2002-213005

[ST.10/C]:

[JP2002-213005]

出 願 人

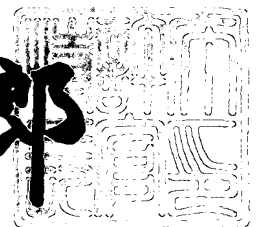
Applicant(s):

オークマ株式会社

2003年 4月25日

特 許 庁 長 官
Commissioner,
Japan Patent Office

太田信一郎



出証番号 出証特2003-3030100

【書類名】 特許願

【整理番号】 3M1-0537

【提出日】 平成14年 7月22日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 11/00

【発明者】

 【住所又は居所】 愛知県丹羽郡大口町下小口5丁目25番地の1 オーク
マ株式会社大口工場内

 【氏名】 大山 博司

【発明者】

 【住所又は居所】 愛知県豊橋市北山町東浦2-1 高師住宅4-403

 【氏名】 高田 広章

【発明者】

 【住所又は居所】 愛知県丹羽郡大口町下小口5丁目25番地の1 オーク
マ株式会社大口工場内

 【氏名】 福島 盛雄

【特許出願人】

 【識別番号】 000149066

 【氏名又は名称】 オークマ株式会社

【代理人】

 【識別番号】 100075258

 【弁理士】

 【氏名又は名称】 吉田 研二

 【電話番号】 0422-21-2340

【選任した代理人】

 【識別番号】 100096976

 【弁理士】

 【氏名又は名称】 石田 純

 【電話番号】 0422-21-2340

【手数料の表示】

【予納台帳番号】 001753

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 論理回路のインターフェース方法およびインターフェースを備えた装置

【特許請求の範囲】

【請求項 1】 論理演算素子により構成される論理回路において、インターフェース定義言語を用いてインターフェースを定義し、前記インターフェース定義言語はソフトウェアオブジェクトに対するインターフェース定義言語と共通部分を有し、前記インターフェース定義言語は機能ごとに機能名および該機能に対する引数および戻り値を宣言する手段を具備し、前記インターフェースを実現するサーバーインターフェース回路には前記インターフェース定義言語に記述された前記機能名を識別する入力および引数の入出力を行う手段および戻り値の出力を行う手段のうち少なくとも前記機能名を識別する入力を行う手段を具備させることを特徴とする論理回路のインターフェース方法。

【請求項 2】 論理演算素子により構成される論理回路において、インターフェース定義言語を用いてインターフェースを定義し、前記インターフェース定義言語はソフトウェアオブジェクトに対するインターフェース定義言語と共通部分を有し、前記インターフェース定義言語は機能ごとに機能名および該機能に対する引数および戻り値を宣言する手段を具備し、前記インターフェースを実現するクライアントインターフェース回路には前記インターフェース定義言語に記述された前記機能名を識別する出力および引数の入出力を行う手段および戻り値の入力を行う手段のうち少なくとも前記機能名を識別する出力を行う手段を具備させることを特徴とする論理回路のインターフェース方法。

【請求項 3】 論理演算素子により構成される論理回路において、インターフェース定義言語を用いてインターフェースを定義し、前記インターフェース定義言語はソフトウェアオブジェクトに対するインターフェース定義言語と共通部分を有し、前記インターフェース定義言語は機能ごとに機能名および該機能に対する引数および戻り値を宣言する手段を具備し、

前記インターフェースを実現するサーバーインターフェース回路として前記インターフェース定義言語に記述された前記機能名を識別する入力および引数の入

出力を行う手段および戻り値の出力を行う手段のうち少なくとも前記機能名を識別する入力を行う手段を具備したサーバー側論理回路と、前記インターフェースを実現するクライアントインターフェース回路として前記機能名を識別する出力および引数の入出力を行う手段および戻り値の入力を行う手段のうち少なくとも前記機能名を識別する出力を行う手段を具備するクライアント側論理回路とを備え、

前記サーバー側論理回路と前記クライアント側論理回路とに関し、前記機能名を識別する出力を行う手段から前記機能名を識別する入力を行う手段へのデータ転送を可能とし、且つ、前記引数を入出力する手段と前記引数を出入力する手段とが互いに備わっていれば前記引数を入出力する手段と前記引数を出入力する手段とのデータ転送を可能とし、且つ、前記戻り値を出力する手段と前記戻り値を入力する手段とが互いに備わっていれば前記戻り値を出力する手段から前記戻り値を入力する手段へのデータ転送を可能とする論理回路を合成することを可能とすることを特徴とする論理回路のインターフェース方法。

【請求項4】 論理演算素子により構成される論理回路において、インターフェース定義言語を用いてインターフェースを定義し、前記インターフェース定義言語はソフトウェアオブジェクトに対するインターフェース定義言語と共通部分を有し、前記インターフェース定義言語は機能ごとに機能名および該機能に対する引数および戻り値を宣言する手段を具備し、前記インターフェースを実現するサーバーインターフェース回路には前記インターフェース定義言語に記述された前記機能名を識別する入力および引数の入出力を行う手段および戻り値の出力を行う手段のうち、少なくとも前記機能名を識別する入力を行う手段を具備することを特徴とするインターフェースを備えた装置。

【請求項5】 論理回路において、インターフェース定義言語を用いてインターフェースを定義し、前記インターフェース定義言語はソフトウェアオブジェクトに対するインターフェース定義言語と共通部分を有し、前記インターフェース定義言語は機能ごとに機能名および該機能に対する引数および戻り値を宣言する手段を具備し、前記インターフェースを実現するクライアントインターフェース回路には前記インターフェース定義言語に記述された前記機能名を識別する出

力および引数の入出力を行う手段および戻り値の入力を行う手段のうち、少なくとも前記機能名を識別する出力を行う手段を具備することを特徴とするインターフェースを備えた装置。

【請求項6】 請求項5に記載のインターフェースを備えた装置において、前記クライアントインターフェース回路の接続端子は、前記サーバーインターフェース回路の接続端子とシステムバスとのいずれとも接続可能とし、前記サーバーインターフェース回路の接続端子へ接続した場合には前記サーバーインターフェース回路を介して装置を駆動可能とし、システムバスに接続した場合には中央処理装置は前記クライアントインターフェース内部のレジスタの値を読み書き可能とし、前記中央処理装置は前記サーバーインターフェース回路を介する装置の機能を代替することを可能とするインターフェースを具備することを特徴とするインターフェースを備えた装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、AND/OR/NOT等の各種ゲートやフリップフロップ等の論理演算素子により構成される論理回路のインターフェース方法およびインターフェースを備えた装置に関し、特に、ソフトウェアオブジェクトとハードウェアオブジェクト（装置）の扱いを統一可能とするものである。

【0002】

【従来の技術】

従来、マイクロプロセッサと周辺LSIとのインターフェースにおいて、状態レジスタやコマンドレジスタなどと呼ばれるレジスタを介して、状態値を読み出したり、所定の処理の実行を制御したりするインターフェース方法が用いられている。

【0003】

このため、周辺LSIを制御するプログラムを設計するプログラマーは、周辺LSIのインターフェースを理解するために、レジスタの意味や複数のレジスタ間の関連性を理解し、レジスタの設定手順に留意してプログラムを作成する必要

がある。

【 0 0 0 4 】

【発明が解決しようとする課題】

ところで、近年の周辺 L S I は、その機能が複雑化しており、多数の状態レジスタやコマンドレジスタを備えている。

【 0 0 0 5 】

従って、レジスタの意味や関連性を容易に理解することが困難となっており、プログラムの誤りを生じさせる虞があるばかりでなく、正誤検出にも時間を要するという問題が生じていた。

【 0 0 0 6 】

また、機能が複雑化しているが故に、その機能をソフトウェア上で実現するかハードウェア上で実現するかの判断も困難となっており、これらが容易に交換可能となる仕組みが必要となっているのが実情である。

【 0 0 0 7 】

一方、近年では、インターフェース定義言語（IDL=Interface Definition Language）によりソフトウェアオブジェクトとのインターフェースを定義して活用する手法が普及している（例えば、OMG（Object Management Group）によるCORBAのIDLや、マイクロソフト社のCOM（Component Object Model）のIDLが知られている）。

【 0 0 0 8 】

しかしながら、これらはソフトウェアオブジェクトを対象としていて、ハードウェアオブジェクト（装置）とのインターフェースは考慮されていないという問題が生じている。

【 0 0 0 9 】

そこで、本願出願人は、これらのインターフェース定義言語を用いてハードウェアとのインターフェースを定義することができれば、ソフトウェアとハードウェアを統一的に扱うことができ、よってソフトウェアオブジェクトとハードウェアオブジェクトを容易に交換することが可能になるなど、プログラマーの負担を

著しく減少させることができ、しかも、ソフトウェアとのインターフェース方法の定義がより簡単なものとなってハードウェア設計者の負担も減少させることができるという考えに至った。

【 0 0 1 0 】

本発明は、上述のような従来の問題に鑑みてなされたもので、周辺 L S I のような論理回路により構成される装置のインターフェースを改善し、プログラマーがレジスタの意味や多数のレジスタ間の関連性を理解するための負担を軽減することができる、しかも、ソフトウェアオブジェクトとハードウェアオブジェクトとの扱いを統一可能とし得て、ソフトウェアオブジェクトとハードウェアオブジェクトの交換を容易にすることができる論理回路のインターフェース方法およびインターフェースを備えた装置を提供することを目的とする。

【 0 0 1 1 】

【課題を解決するための手段】

その目的を達成するため、本発明の論理回路のインターフェース方法は、論理演算素子により構成される論理回路において、インターフェース定義言語を用いてインターフェースを定義し、前記インターフェース定義言語はソフトウェアオブジェクトに対するインターフェース定義言語と共通部分を有し、前記インターフェース定義言語は機能ごとに機能名および該機能に対する引数および戻り値を宣言する手段を具備し、前記インターフェースを実現するサーバーインターフェース回路には前記インターフェース定義言語に記述された前記機能名を識別する入力および引数の入出力を行う手段および戻り値の出力を行う手段のうち少なくとも前記機能名を識別する入力を行う手段を具備させることを特徴とする。

【 0 0 1 2 】

また、本発明の論理回路のインターフェース方法は、論理演算素子により構成される論理回路において、インターフェース定義言語を用いてインターフェースを定義し、前記インターフェース定義言語はソフトウェアオブジェクトに対するインターフェース定義言語と共通部分を有し、前記インターフェース定義言語は機能ごとに機能名および該機能に対する引数および戻り値を宣言する手段を具備し、前記インターフェースを実現するクライアントインターフェース回路には前

記インターフェース定義言語に記述された前記機能名を識別する出力および引数の入出力を行う手段および戻り値の入力を行う手段のうち少なくとも前記機能名を識別する出力を行う手段を具備させることを特徴とする。

【 0 0 1 3 】

また、本発明の論理回路のインターフェース方法は、論理演算素子により構成される論理回路において、インターフェース定義言語を用いてインターフェースを定義し、前記インターフェース定義言語はソフトウェアオブジェクトに対するインターフェース定義言語と共通部分を有し、前記インターフェース定義言語は機能ごとに機能名および該機能に対する引数および戻り値を宣言する手段を具備し、前記インターフェースを実現するサーバーインターフェース回路として前記インターフェース定義言語に記述された前記機能名を識別する入力および引数の入出力を行う手段および戻り値の出力を行う手段のうち少なくとも前記機能名を識別する入力を行う手段を具備したサーバー側論理回路と、前記インターフェースを実現するクライアントインターフェース回路として前記機能名を識別する出力および引数の入出力を行う手段および戻り値の入力を行う手段のうち少なくとも前記機能名を識別する出力を行う手段を具備するクライアント側論理回路とを備え、前記サーバー側論理回路と前記クライアント側論理回路とに関し、前記機能名を識別する出力を行う手段から前記機能名を識別する入力を行う手段へのデータ転送を可能とし、且つ、前記引数を入出力する手段と前記引数を出入力する手段とが互いに備わっていれば前記引数を入出力する手段と前記引数を出入力する手段とのデータ転送を可能とし、且つ、前記戻り値を出力する手段と前記戻り値を入力する手段とが互いに備わっていれば前記戻り値を出力する手段から前記戻り値を入力する手段へのデータ転送を可能とする論理回路を合成することを可能とすることを特徴とする。

【 0 0 1 4 】

さらに、本発明のインターフェースを備えた装置は、論理演算素子により構成される論理回路において、インターフェース定義言語を用いてインターフェースを定義し、前記インターフェース定義言語はソフトウェアオブジェクトに対するインターフェース定義言語と共通部分を有し、前記インターフェース定義言語は

機能ごとに機能名および該機能に対する引数および戻り値を宣言する手段を具備し、前記インターフェースを実現するサーバーインターフェース回路には前記インターフェース定義言語に記述された前記機能名を識別する入力および引数の入出力を行う手段および戻り値の出力を行う手段のうち、少なくとも前記機能名を識別する入力を行う手段を具備することを特徴とする。

【 0 0 1 5 】

また、本発明のインターフェースを備えた装置は、論理回路において、インターフェース定義言語を用いてインターフェースを定義し、前記インターフェース定義言語はソフトウェアオブジェクトに対するインターフェース定義言語と共通部分を有し、前記インターフェース定義言語は機能ごとに機能名および該機能に対する引数および戻り値を宣言する手段を具備し、前記インターフェースを実現するクライアントインターフェース回路には前記インターフェース定義言語に記述された前記機能名を識別する出力および引数の入出力を行う手段および戻り値の入力を行う手段のうち、少なくとも前記機能名を識別する出力を行う手段を具備することを特徴とする。

【 0 0 1 6 】

また、前記クライアントインターフェース回路の接続端子は、前記サーバーインターフェース回路の接続端子とシステムバスとのいずれとも接続可能とし、前記サーバーインターフェース回路の接続端子へ接続した場合には前記サーバーインターフェース回路を介して装置を駆動可能とし、システムバスに接続した場合には中央処理装置は前記クライアントインターフェース内部のレジスタの値を読み書き可能とし、前記中央処理装置は前記サーバーインターフェース回路を介する装置の機能を代替することを可能とするインターフェースを具備することを特徴とする。

【 0 0 1 7 】

【発明の実施の形態】

次に、本発明の実施の形態を図面に基づいて説明する。

【 0 0 1 8 】

図 1 は、本発明のコンピュータシステムの構成図である。この図 1 において、

1はCPU（中央処理装置）、2はメモリ、3は乱数発生装置である。このメモリ2と乱数発生装置3とはシステムバス4を介してCPU1と接続されている。また、CPU1はメモリ2上に記憶されるプログラムを解釈・処理する。

【0019】

図2は乱数発生装置3のブロック構成図である。乱数発生装置3は、IDLサーバーインターフェース部11を有する。

【0020】

このIDLサーバーインターフェース部11は、機能識別レジスタ12、引数レジスタ13、戻り値レジスタ14、状態レジスタ15を備え、外部端子16によりシステムバス4に接続される。尚、図2では、各レジスタ12～15とシステムバス4とのデータの授受を制御する部分は省略されているが、システムバス4に含まれるアドレスバスや制御バスにより各レジスタ12～15とのデータの受渡しが制御される。

【0021】

図3はIDLによるインターフェースの定義例の説明図である。この例は、上述したCORBAのIDLとの共通部分があり、インターフェースの名称、機能名、機能名ごとの引数および戻り値が記してある。また、図3の例では、インターフェース名をrandomGeneratorとする乱数発生装置3のインターフェースを定義したもので、乱数の種を設定する機能（機能名：setSeed）と、発生した乱数を得る機能（機能名：getRandom）が示されている。そして、setSeedには入力引数としてdouble型の種（seed）が渡されることを表している。また、getRandomは戻り値として発生した乱数をdouble型の値として返すことを表している。

【0022】

図4および図5は、図3に例示したインターフェース定義を元に、本発明によるインターフェースを備えた装置をCPU1により操作するために作成したC言語のヘッダファイルおよびプログラムの説明図である。

【0023】

このようなヘッダファイルやプログラムは、インターフェース定義から一定の

規則に基づいて変換して作成できるため、通常IDLコンパイラと呼ばれるプログラムにより変換を行う。CORBAなどのIDLコンパイラはインターフェース定義を通信データに変換するCやC++などのヘッダファイルやプログラムを生成するが、ここでは乱数発生装置3のIDLサーバーインターフェース部11のレジスタ設定または参照を行うCのプログラムが生成されるよう、IDLコンパイラが構成される。なお、このような役割を果たすプログラムは、スタブあるいはプロキシなどと呼ばれる。

【 0 0 2 4 】

図5では、インターフェース定義されたsetSeedとgetRandomに対応する関数『setSeed』と『getRandom』とが記述されている。

【 0 0 2 5 】

本発明の方法あるいは装置は複数部の協調により達成される。図1および図2に示した構成図をもとに、これらの関数および乱数発生装置3の動作の概要を説明する。

【 0 0 2 6 】

CPU1はメモリ2に格納されているプログラムを読み出して処理を行う。そのプログラム内で関数『setSeed』が呼び出されると、関数『setSeed』のプログラムに基づき、CPU1は機能識別の値『FID__setSeed』を乱数発生装置3の機能識別レジスタ12に書き込む。

【 0 0 2 7 】

この定数『FID__setSeed』は図4のヘッダファイルにて定義されているもので、以下の『FID__getRandom』や『Finished』などの定数も同様である。続いてdouble型の引数『seed』を引数レジスタ13に書き込み、この関数の処理は終了する。

【 0 0 2 8 】

乱数発生装置3のIDLサーバーインターフェース部11は、機能setSeedに対するすべての引数を受け取ったことを確認すると、直ちに動作を開始させる。この例の場合、乱数発生部17の乱数の種レジスタ18に乱数の種を格納

する動作が行われる。

【 0 0 2 9 】

次に関数『getRandom』が呼び出されると、機能識別の値『FID_getRandom』を乱数発生装置3の機能識別レジスタ12に書き込む。IDLサーバーインターフェース部11は、機能『getRandom』に引数がないことから、直ちに乱数発生部に乱数の発生をさせる。IDLサーバーインターフェース部11は乱数発生部17の乱数の発生が完了すると、それを戻り値レジスタ14に書き込む。関数『getRandom』では、戻り値レジスタ14に値が書き込まれるまで、関数『getStatus』を呼び出して状態レジスタ15の値を読み取り、その値が『Finished』となるまでwhile文により待ち続ける。以上、本発明の装置を組み込んだコンピュータシステムの全体の動作の概要を説明した。

【 0 0 3 0 】

次に、図7のフロー図を用いて図1に示したコンピュータシステムの設計作業の流れを説明する。尚、以下のフロー図並びにその説明では、各ステップを記号「S」で省略する。

【 0 0 3 1 】

設計作業は、最初にインターフェースを決定する（S21）。インターフェースの決定では機能をすべて挙げ、各機能ごとに機能名、引数、戻り値をそれぞれ決定する。

【 0 0 3 2 】

次に、S21で決定したインターフェースを図3のインターフェース定義ファイルとして作成する（S22）。そして、インターフェース定義ファイルをIDLコンパイラに通してヘッダファイルとスタブを作成する（S23）。IDLコンパイラにより図4のヘッダファイルと図5のスタブが生成される。

【 0 0 3 3 】

ここでソフトウェア設計とハードウェア設計に分かれる。

【 0 0 3 4 】

ソフトウェア設計では、次にスタブをコンパイラによりコンパイルしてリロ

ケータブルオブジェクトを作成する（S 2 4）。次に他の部分のリロケータブルオブジェクトとリンクして実行モジュールを作成し（S 2 5）、ソフトウェアの設計作業は完了する。

【 0 0 3 5 】

ハードウェア設計者は、図 3 のインターフェース定義ファイルと図 4 のヘッダファイルを参照しながら I D L サーバーインターフェース部 1 1 を設計する（S 2 6）。ここで参照するインターフェース定義ファイルとヘッダファイルの参照する内容については、I D L サーバーインターフェース部 1 1 の動作の詳細とともに説明する。

【 0 0 3 6 】

次にハードウェア設計者は、乱数発生部 1 7 の回路と I D L サーバーインターフェース部 1 1 を結合する回路を設計して（S 2 7）、乱数発生装置 3 の設計作業は完了する。

【 0 0 3 7 】

引き続き、C P U 1、メモリ 2、乱数発生装置 3、システムバス 4 を接続する回路の設計作業を行い（S 2 8）、ハードウェア設計作業は終了する。

【 0 0 3 8 】

次に、図 6 の I D L サーバーインターフェース部 1 1 の動作を表すフローチャートを用いて、本発明の中心である I D L サーバーインターフェース部 1 1 の動作を、もう一度詳しく説明する。

【 0 0 3 9 】

I D L サーバーインターフェース部 1 1 は、機能識別レジスタ 1 2 に外部端子 1 6 を介して値が書き込まれるのを待つ（S 1）。

【 0 0 4 0 】

ここで、機能識別レジスタ 1 2 に書き込まれる値は、`setSeed`機能であれば、これに対応する値は、図 4 のヘッダファイルにおいて`#define`により定義される『`FID_setSeed`』の値であり、具体的な値は『1』である。また、`getRandom`機能であれば『`FID_getRandom`』であり、具体的な値は『2』である。

【 0 0 4 1 】

機能識別レジスタ 1 2 に値が書き込まれると、状態レジスタ 1 5 の値を実行中を意味する値『E x e c u t i n g』に変更し（S 2）、機能識別レジスタ 1 2 の内容から引数の個数を割り出す（S 3）。ここで、引数の個数の割り出しは、引数個数算出部 2 0 により行われるが、引数個数算出部 2 0 には図 3 のインターフェース定義ファイルおよび図 4 のヘッダを参照して、予め機能識別と引数個数との対応関係を引数個数算出部 2 0 に記憶するように設計しておく。

【 0 0 4 2 】

次に、引数がすべて入力されているかどうかを判定し（S 4）、受け取っていない引数があれば引数レジスタ 1 3 に引数が 1 つ書き込まれるのを待ち（S 5）、再び S 4 から繰り返す。もしすべての引数が入力されていれば（引数がない場合は、すべて入力されたものとみなす）、機能識別により指定された機能を開始させる（S 6）。I D L サーバーインターフェース部 1 1 の設計者は、図 4 のヘッダファイルを参照して機能識別と対応する機能を一致させておく。

【 0 0 4 3 】

次に、機能処理の終了を待つ（S 7）。その機能の処理が終了すると、戻り値があるかどうかを判定して（S 8）、戻り値があれば戻り値を戻り値レジスタ 1 4 に書き込んで（S 9）、最後に状態レジスタ 1 5 の値を『F i n i s h e d』に変更し、再び S 1 から繰り返す。

【 0 0 4 4 】

以上の説明において、図 5 では、状態レジスタが『F i n i s h e d』に変化するまで、w h i l e ループで待つようになっているが、状態レジスタ 1 5 の値が変化したときに C P U 1 へ割り込みを発生するようにして、それまでこのプログラムが待機するように変更すれば C P U 1 の利用効率を上げることができる。また、引数の個数やデータ型の検査を加えるなどの変更を行ってもよい。また、インターフェース定義において、すべての機能が引数を持たないのであれば、引数レジスタ 1 3 を省略し、図 6 で示したルーチンの S 4, S 5 を省略することもできる。また、全ての機能が戻り値を返さないのであれば、戻り値レジスタ 1 4 および図 6 で示したルーチンの S 8, S 9 を省略することもできる。また、乱数

発生装置 3 の働きをソフトウェアによりエミュレートするように変更する場合、図 5 の関数『setSeed』および関数『getRandom』を置き換えればよく、ハードウェアオブジェクトを容易にソフトウェアオブジェクトに交換することが可能である。これは、本発明による方法および装置で用いるインターフェース定義言語がソフトウェアオブジェクトを対象とするインターフェース定義言語と共通部分があるために、容易に交換が可能となる。

【 0 0 4 5 】

次に、図 2 に示した乱数発生装置 3 を利用して、モンテカルロ法により円周率を計算する円周率計算装置 5 の構成図を図 1 0 に示す。また、この円周率計算装置 (piCalculator) のインターフェース定義を図 9 に示す。この円周率計算装置 5 を組み込んだコンピュータシステムの構成図を図 8 に示す。

【 0 0 4 6 】

図 1 0 の円周率計算装置 5 は、IDL サーバーインターフェース部 2 1、円周率計算部 2 7、IDL クライアントインターフェース部 3 1 からなり、IDL サーバーインターフェース部 2 1 に接続された外部端子 2 6 はシステムバス 4 に接続され、IDL クライアントインターフェース部 3 1 に接続された外部端子 3 6 は乱数発生装置 3 の外部端子 1 6 に接続される。

【 0 0 4 7 】

外部端子 3 6 と外部端子 1 6 の接続によって、機能識別レジスタ 3 2 から機能識別レジスタ 1 2 へのデータ転送、引数レジスタ 3 3 と引数レジスタ 1 3 とのデータ転送、戻り値レジスタ 1 4 から戻り値レジスタ 3 4 へのデータ転送が可能となり、円周率計算装置 5 から乱数発生装置 3 に対する機能の要求が可能となる。

【 0 0 4 8 】

図 8 のコンピュータシステムにおいて、乱数発生装置 3 を使用しないように変更したコンピュータシステムの構成図が図 1 1 である。図 1 1 では、円周率計算装置 5 の IDL クライアントインターフェース部 3 1 に接続された外部端子 3 6 はシステムバス 4 に接続される。

【 0 0 4 9 】

コンピュータシステムを図 8 の構成から図 1 0 の構成に変更可能とするために

、円周率発生装置 5 の I D L クライアントインターフェース部 3 1 は、乱数発生装置 3 の I D L サーバーインターフェース部 1 1 を駆動可能とするとともに、C P U 1 から機能識別レジスタ 3 2、引数レジスタ 3 3、戻り値レジスタ 3 4、状態レジスタ 3 5 を読み出しまたは書き込み可能とすることで達成される。

【 0 0 5 0 】

この時、C P U 1 は、円周率発生装置 5 の I D L クライアントインターフェース部 3 1 から発せされる要求を監視して、機能識別に応じた機能の処理を行う必要がある。

【 0 0 5 1 】

図 1 2 は、C P U 1 が行う、要求の監視、機能の処理を説明するフローチャートである。

【 0 0 5 2 】

まず、円周率計算装置 5 の I D L クライアントインターフェース部 3 1 の状態レジスタ 3 5 の値を読み出す (S 3 1) 。その値が『 R e q u e s t i n g 』であれば、機能の処理の要求があると判断し (S 3 2) 、機能識別レジスタ 3 2 の値を読み出す (S 3 3) 。

【 0 0 5 3 】

機能識別レジスタ 3 2 の値が『 F I D _ s e t S e e d 』であれば (S 3 4) 、引数レジスタ 3 3 の値を読み出し、乱数の種として記憶する (S 3 5) 。

【 0 0 5 4 】

機能識別レジスタ 3 2 の値が『 F I D _ g e t R a n d o m 』であれば (S 3 6) 、乱数を発生し (S 3 7) 、その乱数を戻り値レジスタ 3 4 へ書き込む (S 3 8) 。

【 0 0 5 5 】

上述した S 3 5、S 3 8 が終了すれば、再び S 3 1 へ戻る。このプログラムは C P U 1 を占有してしまうので、タスクの 1 つとして実装する必要がある。あるいは、状態レジスタ 3 5 の値が『 R e q u e s t i n g 』に変化したときに、C P U 1 に割り込みが発生するようにして、それまで S 3 1 で待機するようにすれば、C P U 1 の利用効率を向上することができる。

【 0 0 5 6 】

図 1 3、図 1 4 は図 1 2 の処理を C 言語化したヘッダファイルおよびプログラムである。インターフェース定義から一定の規則に従って変換できるため、先述の図 4、図 5 のヘッダファイル、プログラムファイルと同様に I D L コンパイラと呼ばれるプログラムを使って作成することができる。なお、このような役割を果たすプログラムは、スケルトンあるいはスタブなどと呼ばれる。

【 0 0 5 7 】

図 1 5 は、上述のような特性を持った I D L クライアントインターフェース部 3 1 の動作を表すフローチャートである。始めに円周率計算部 2 7 から機能要求がないかを判断し (S 4 1)、もしあれば機能識別レジスタ 3 2 に機能要求に対応する機能識別を書き込む (S 4 2)。ここでは、『 F I D _ s e t S e e d 』か『 F I D _ g e t R a n d o m 』のいずれかの値が書き込まれる。次に機能要求が s e t S e e d であるかを判断し (S 4 3)、s e t S e e d であれば引数が存在するので、円周率計算部 2 7 から引数を受け取り引数レジスタ 3 3 に書き込み (S 4 4)、状態レジスタ 3 5 の値を『 R e q u e s t i n g 』へと変更する (S 4 5)。

【 0 0 5 8 】

ここで円周率計算装置 5 の外部端子 3 6 の接続先が乱数発生装置 3 であるか、C P U 1 であるかにより動作が変わる。

【 0 0 5 9 】

接続先が C P U 1 であれば、C P U 1 は先述の図 1 2 のフローチャートで示した処理を行っており、機能識別の値が『 F I D _ s e t S e e d 』であれば、引数は S 3 5 で読み込まれる。機能識別の値が『 F I D _ g e t R a n d o m 』であれば、引数はない。S 4 7 では、すべての引数が読み込まれたかどうかを判断し (引数がない場合は、読み込まれたものと判断する)、すべて読み込まれている場合は、戻り値があるかどうか判断し (S 4 8)、戻り値がある場合は戻り値レジスタ 3 4 に値が書き込まれるまで待つ (S 4 9)。戻り値レジスタ 3 4 に書き込まれた値を乱数として円周率計算部 2 7 に渡す (S 5 0)。

【 0 0 6 0 】

以上でCPU1とのやり取りは終わって、状態レジスタ35の値を『Waiting』に変更し(S51)、再びS41から繰り返す。

【0061】

もし、S46にて、円周率計算装置5の外部端子36の接続先が乱数発生装置3であると判断された場合には、機能識別レジスタ32の内容を接続先である乱数発生装置3のIDLサーバーインターフェース部11の機能識別レジスタ12に書き込む(S52)。

【0062】

ここで乱数発生装置3は図6のフローチャートに従って動作しており、S1において機能識別レジスタ12への書き込みを検出して、S2以降の動作を行う。IDLクライアントインターフェース部31のS53以降の動作は、乱数発生装置3のIDLサーバーインターフェース部11のS4以降の動作と対応付けられている。

【0063】

次に、引数があるかどうかを判断する(S53)。機能識別の値が『FID__setSeed』であれば、引数があると判断される。引数があれば引数レジスタ33の内容を乱数発生装置3の引数レジスタ13に書き込む(S54)。

【0064】

次に、戻り値があるかどうかを判断する(S55)。機能識別の値が『FID__getRandom』であれば戻り値があると判断される。戻り値があると判断された場合、乱数発生装置3の状態レジスタ15の値が『Finished』になるまで待つ(S56)。続いて乱数発生装置3の戻り値レジスタ14の値を読み出し、戻り値レジスタ34に書き込む(S57)。そして、戻り値レジスタ34の内容を円周率計算部27に渡す(S58)。

【0065】

以上で乱数発生装置3とのやり取りは終わって、状態レジスタ35の値を『Waiting』に変更し(S51)、再びS41から繰り返す。

【0066】

ところで、上記実施の形態では、乱数発生装置3と円周率計算装置5を例に取

って説明したが、乱数発生部 17 や、円周率計算部 27 を変更し、またそれにあわせてインターフェース定義 (randomGenerator, piCalculator) を変更することにより、種々の装置に対応できる。

【0067】

また、上記実施の形態では、引数の型や引数の個数の照合を行っていないが、それらを行うことにより、インターフェースプログラムの誤りを厳密に検出することができる。

【0068】

さらに、上記実施の形態では、入力引数の例のみを示したが出力引数あるいは入出力引数の扱いは可能で、出力引数の場合、入力引数の場合とはデータ転送の向きが逆になる。また、本実施例ではインターフェース記述言語として CORBA の IDL を例に説明したが、ソフトウェアオブジェクトを対象とする IDL と共通部分があり、機能名および該機能に対する引数および戻り値を宣言する手段を備えたものであれば CORBA の IDL に限定されるものではない。

【0069】

また、上記実施の形態のプログラムではインスタンスオブジェクトを区別するための記述を省略して簡略化しているが、インスタンスオブジェクトを区別して取り扱う手段を備えることが望ましい。

【0070】

尚、本発明の方法及び装置は、集積回路であって外部とのインターフェースを目的とするのがもっとも効果的であるが、集積回路内部のブロック間のインターフェースあるいは複数の集積回路からなる論理回路間のインターフェースに用いてもよい。

【0071】

【発明の効果】

本発明の方法および装置によれば、ソフトウェアオブジェクトと共通部のあるインターフェース定義言語によりハードウェアとのインターフェースを定義することができるから、ソフトウェアとハードウェアを統一的に扱うことが可能になり、ソフトウェアオブジェクトとハードウェアオブジェクト（装置）を容易に交

換することが可能になるなどプログラマーの負担を著しく減少させることができる。また、ソフトウェアとのインターフェース方法の定義がより簡単なものになり、ハードウェア設計者の負担を減少することができる。また、ハードウェアとソフトウェアの並行開発が容易になり、開発納期の短縮を図ることができる。ハードウェアオブジェクト同士の接続と中央処理装置への接続とを容易に変更できるので、ハードウェア処理していたものをソフトウェア処理に置き換える、あるいはその逆が容易になるなど絶大な効果を有する。

【図面の簡単な説明】

【図 1】 本発明のコンピュータシステムの構成図である。

【図 2】 乱数発生装置のブロック構成図である。

【図 3】 インターフェース定義言語（IDL）による乱数発生装置のインターフェース定義の説明図である。

【図 4】 図 5 のプログラムに対応するヘッダファイルの説明図である。

【図 5】 乱数発生装置を直接操作するためのプログラムの説明図である。

【図 6】 IDL サーバーインターフェース部の動作を表すフロー図である。

。

【図 7】 設計作業の流れを表すフロー図である。

【図 8】 円周率計算装置を組み込んだコンピュータシステムのブロック構成図である。

【図 9】 円周率計算装置（`piCalculator`）のインターフェース定義の説明図である。

【図 10】 円周率を計算する円周率計算装置のブロック構成図である。

【図 11】 乱数発生装置を使用しないように変更したコンピュータシステムのブロック構成図である。

【図 12】 中央処理装置が行う、要求の監視及び機能の処理に関するフロー図である。

【図 13】 図 14 のプログラムに対応するヘッダファイルの説明図である。

。

【図 14】 図 12 の処理を C 言語化したプログラムの説明図である。

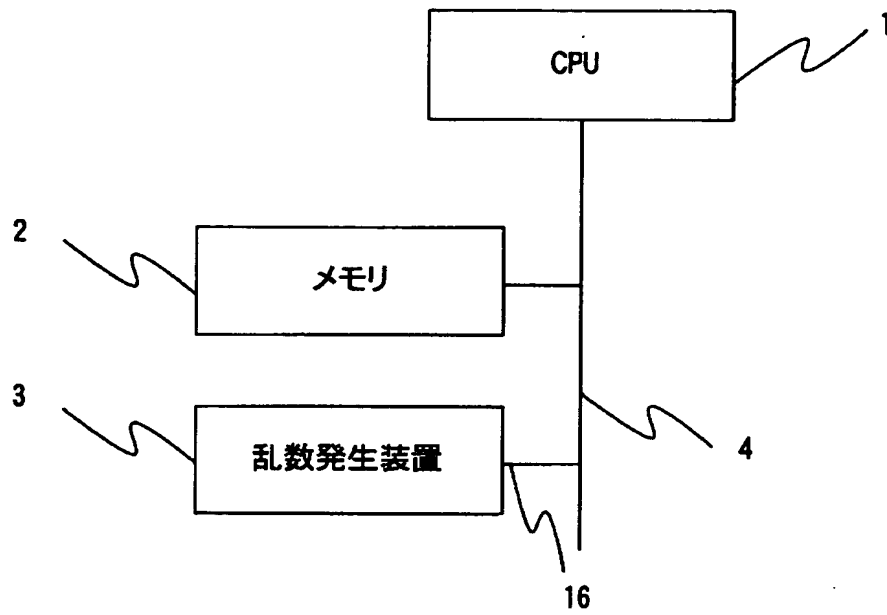
【図 1 5】 I D L クライアントインターフェース部の動作を表すフロー図である。

【符号の説明】

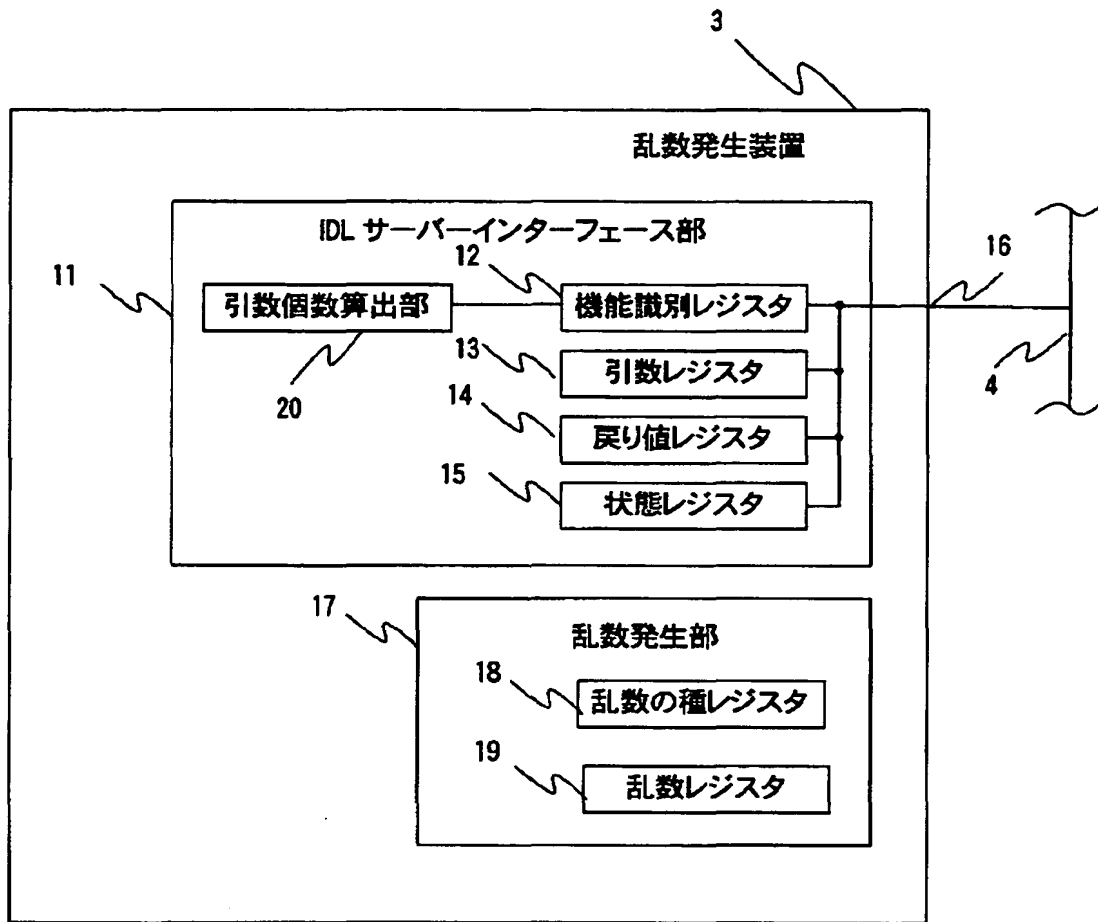
1 中央制御装置、2 メモリ、3 乱数発生装置、4 システムバス、5 円周率計算装置、1 1 (乱数発生装置の) I D L サーバーインターフェース部、1 2 機能識別レジスタ、1 3 引数レジスタ、1 4 戻り値レジスタ、1 5 状態レジスタ、1 6 外部端子、1 7 乱数発生部、1 8 乱数の種レジスタ、1 9 乱数レジスタ、2 1 (円周率計算装置の) I D L サーバーインターフェース部、2 2 機能識別レジスタ、2 3 引数レジスタ、2 4 戻り値レジスタ、2 5 状態レジスタ、2 6 外部端子、2 7 円周率計算部、2 8 回数レジスタ、2 9 円周率レジスタ、3 1 I D L クライアントインターフェース部、3 2 機能識別レジスタ、3 3 引数レジスタ、3 4 戻り値レジスタ、3 5 状態レジスタ、3 6 外部端子。

【書類名】 図面

【図 1】



【図 2】



【図 3】

```
interface randomGenerator{
    void setSeed( in double seed );
    double getRandom();
};
```


【図 4】

```

/* IDL インターフェースに共通の部分 */
/* 状態レジスタの値 */
#define Executing 1
#define Finished 2
#define Requesting 3

/* レジスタにアクセスする関数 */
void putFunctionID( FID );
void putDoubleArg( double );
double getDoubleReturn();
int getStatus();

/* randomGenerator 固有部 */
/* 機能識別の値の定義 */
#define FID_setSeed 1
#define FID_getRandom 2

/* 機能に対応する C 言語関数のプロトタイプ宣言 */
void setSeed( double seed );
double getRandom()

```

【図 5】

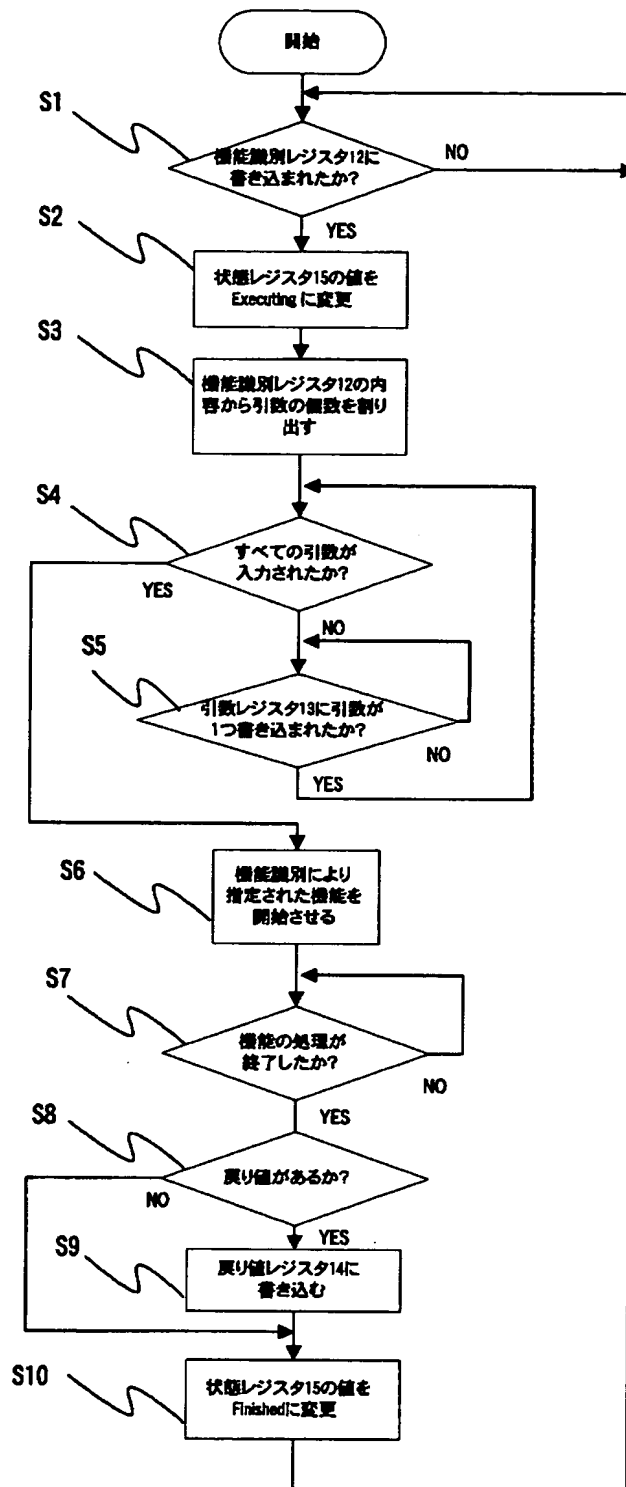
```

void setSeed( double seed )
{
    putFunctionID( FID_setSeed );
    putDoubleArg( seed );
}

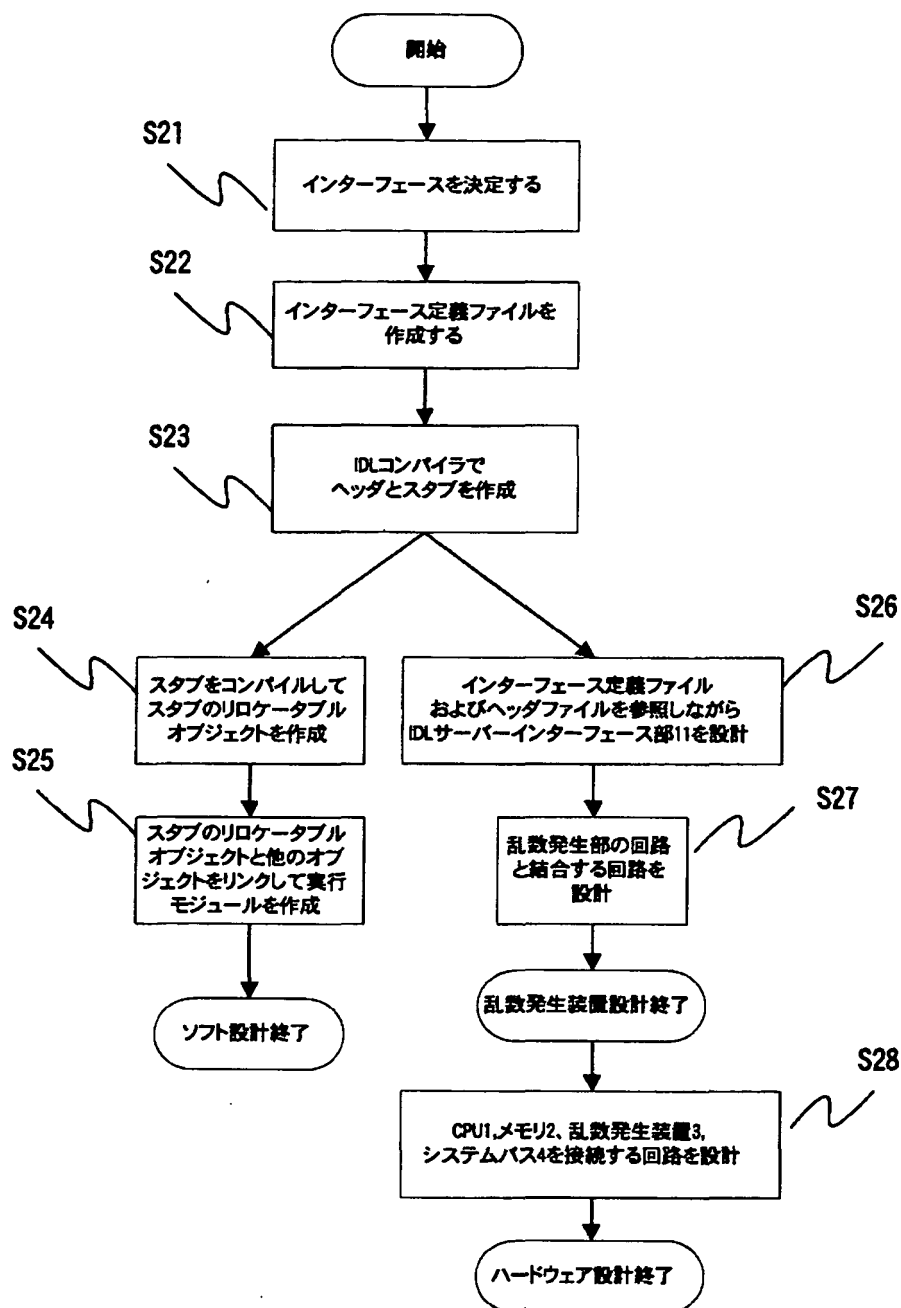
double getRandom()
{
    putFunctionID( FID_getRandom );
    while( getStatus() != Finished )
        ;
    return getDoubleReturn();
}

```

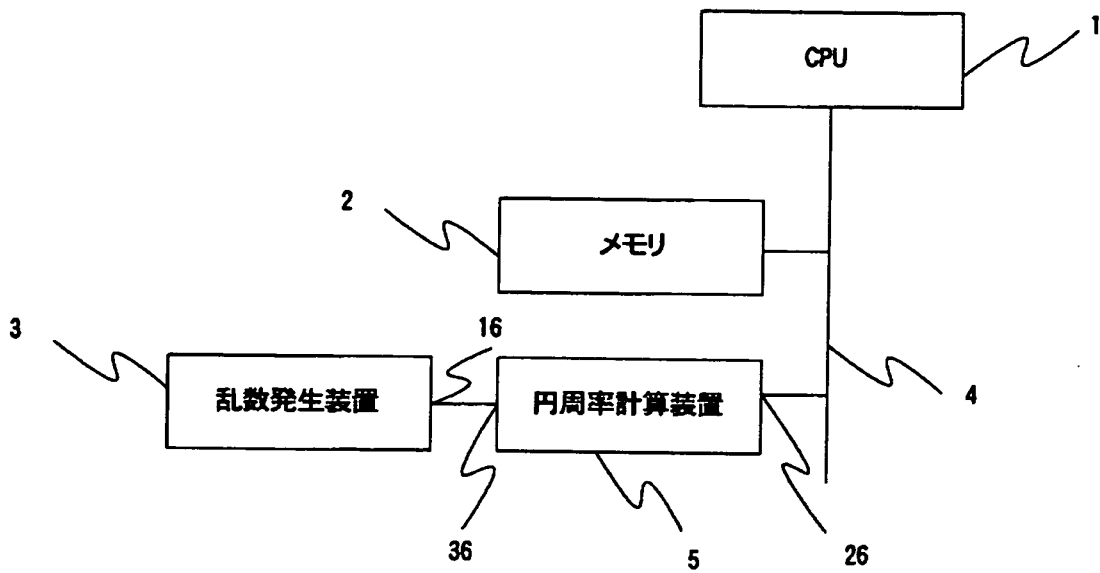
【図 6】



【図 7】



【図 8】



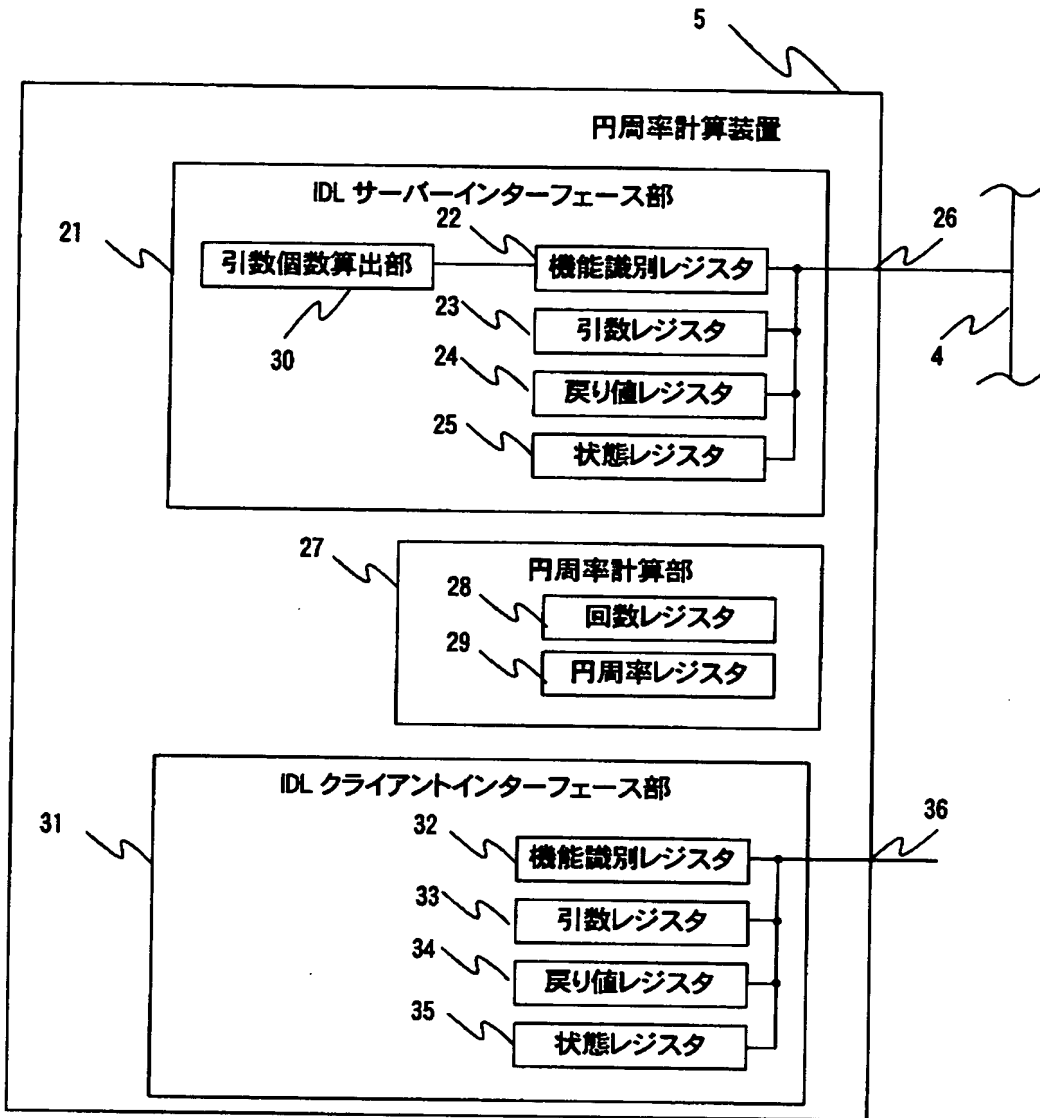
【図 9】

```

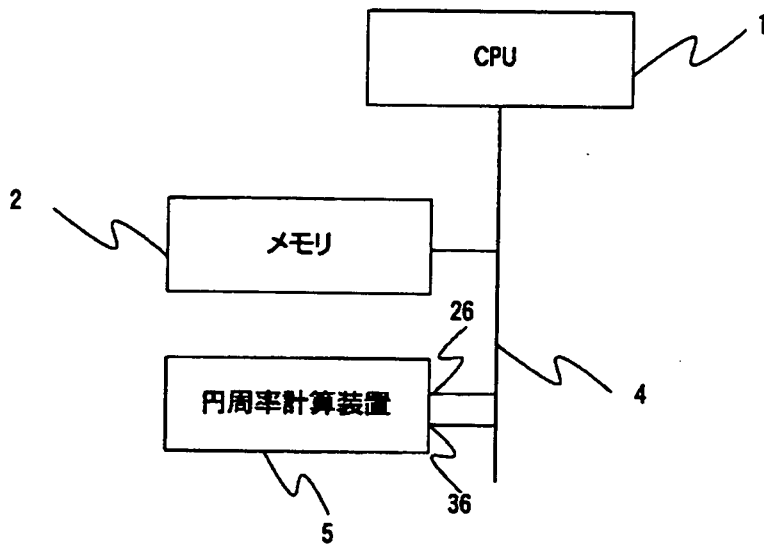
interface piCalculator{
    void setCount( in int count );
    double getPi();
};

```

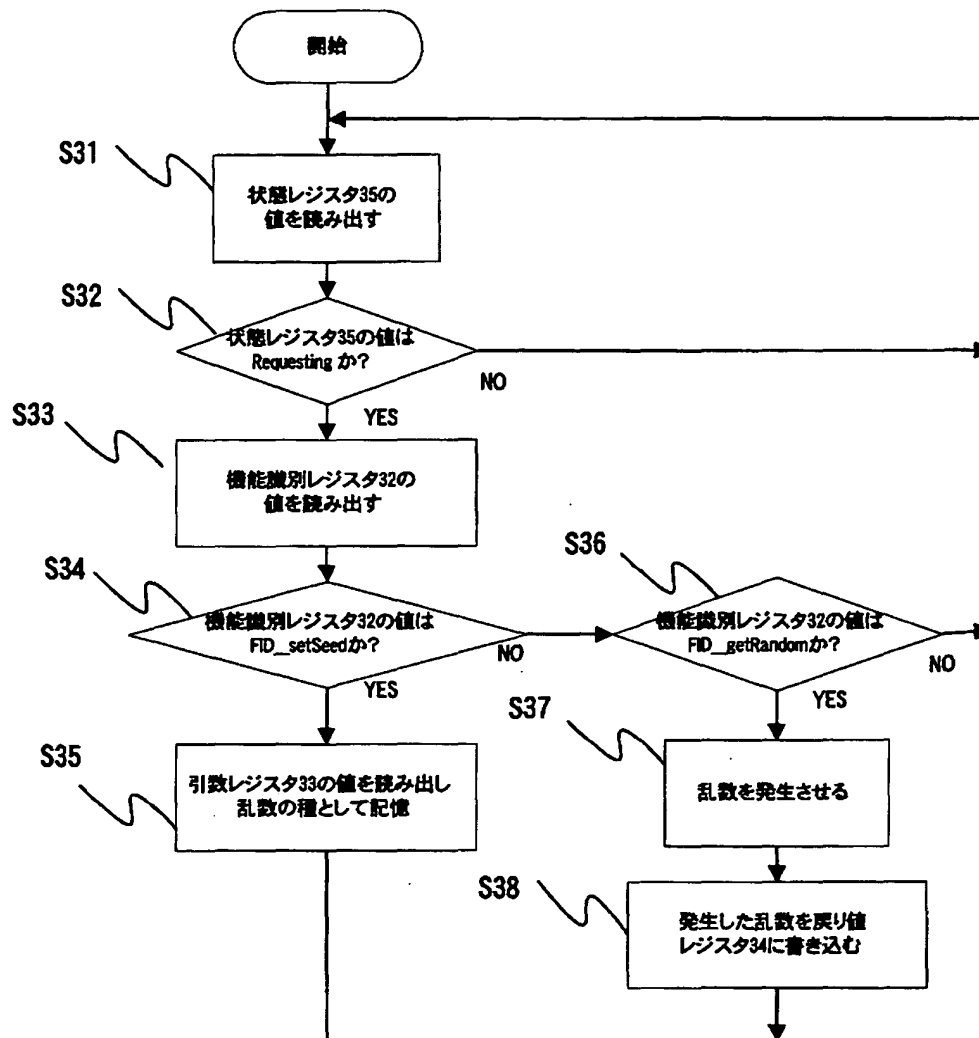
【図10】



【図 1 1】



【図 12】



【図 1 3】

```

/* IDL インターフェースに共通の部分 */
/* 状態レジスタの値 */
#define Waiting 0
#define Executing 1
#define Finished 2
#define Requesting 3

/* レジスタにアクセスする関数 */
FID getFunctionID();
double getDoubleArg();
void putDoubleReturn( double );
int getStatus();

/* randomGenerator 固有部 */
/* 機能識別の値の定義 */
#define FID_setSeed 1
#define FID_getRandom 2

/* 機能に対応する C 言語関数のプロトタイプ宣言 */
void setSeed( double seed );
double getRandom()

```

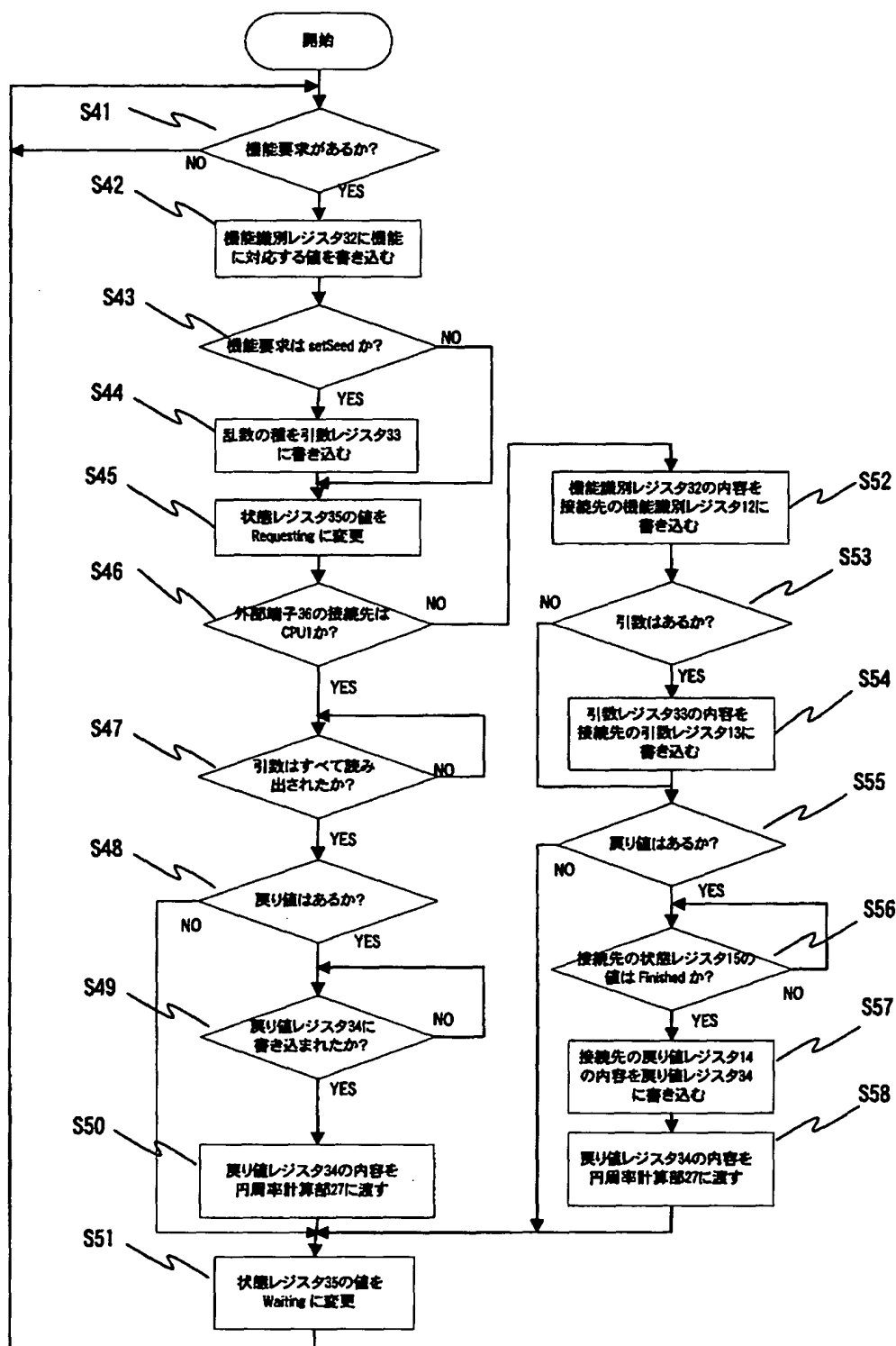
【図 1 4】

```

main()
{
    while(1){
        while( getStatus() != Requesting )
            ;
        switch( getFunctionID() ){
            case FID_setSeed:
                setSeed( getDoubleArg() );
                break;
            case FID_getRandom:
                putDoubleReturn( getRandom() );
                break;
        }
    }
}

```


【図15】



【書類名】 要約書

【要約】

【課題】 プログラマーがレジスタの意味や多数のレジスタ間の関連性を理解するための負担を軽減することができ、しかも、ソフトウェアオブジェクトとハードウェアオブジェクトとの扱いを統一可能とし、ソフトウェアオブジェクトとハードウェアオブジェクトの交換を容易にする。

【解決手段】 インターフェース定義言語を用いてインターフェースを定義し、インターフェース定義言語はソフトウェアオブジェクトに対するインターフェース定義言語と共通部分を有すると共に機能ごとに機能名および機能に対する引数および戻り値を宣言する手段を具備し、インターフェースを実現するサーバーインターフェース回路にはインターフェース定義言語に記述された機能名を識別する入力および引数の入出力を行う手段および戻り値の出力を行う手段のうち少なくとも機能名を識別する入力を行う手段を具備した。

【選択図】 図 2

出 願 人 履 歴 情 報

識別番号 [000149066]

1. 変更年月日	2001年 7月 2日
[変更理由]	住所変更
住 所	愛知県丹羽郡大口町下小口五丁目25番地の1
氏 名	オークマ株式会社